Asynchronous Evolutionary Multi-Objective Algorithms with heterogeneous evaluation costs

Mouadh Yagoubi, Ludovic Thobois, Marc Schoenauer

Asynchronous Evolutionary Multi-Objective Algorithms with Heterogeneous Evaluation Costs

Mouadh Yagoubi*†, Ludovic Thobois* and Marc Schoenauer†
*PSA PEUGEOT CITROËN, Centre technique de Vélizy-Villacoublay, France.
Email: first.last@mpsa.com
†EPC TAO, INRIA Saclay and LRI, Université Paris-Sud, Orsay, France.
Email: first.last@inria.fr

***Abstract*—Master-slave parallelization of Evolutionary Algorithms (EAs) is straightforward, by distributing all fitness computations to slaves. The benefits of asynchronous steady-state approaches are well-known when facing a possible heterogeneity among the evaluation costs in term of runtime, be they due to heterogeneous hardware or non-linear numerical simulations. However, when this heterogeneity depends on some characteristics of the individuals being evaluated, the search might be biased, and some regions of the search space poorly explored. Motivated by a real-world case study of multi-objective optimization problem – the optimization of the combustion in a Diesel Engine – the consequences of different components of heterogeneity in the evaluation costs on the convergence of two Evolutionary Multi-objective Optimization Algorithms are investigated on artificially-heterogeneous benchmark problems. In some cases, better spread of the population on the Pareto front seem to result from the interplay between the heterogeneity at hand and the evolutionary search.**

I. Introduction

Most real-world problems are multi-criteria by nature, involving several contradictory objectives (e.g., typically, cost and quality). Furthermore, those objectives, and the many additional constraints that characterize real-world problems, are generally not regular (e.g., not differentiable), thus prohibiting the use of traditional optimization methods. Finally, those objectives are often noisy and multi-modal. Hence, stochastic methods like Evolutionary Multi-Objective Optimization Algorithms (EMOAs) [1], [2] are good candidates for tackling such problems, thanks to their robustness and their flexibility. Unfortunately, the main drawback of Evolutionary Algorithms (EAs) in general, and EMOAs in particular, is their high cost in terms of number of function evaluations required to reach a satisfactory solution. And this drawback can become prohibitive for those real-world problems where the computation of the objectives is made through heavy numerical simulations that can take hours or even days to complete.

At the same time, computer hardware is experiencing what is sometimes called "the end of Moore law", i.e., the end of the exponential increase of the computing power available in one CPU. The answer to this critical issue is today brought distributed computing, be it on multi-core CPUs, on clusters of CPUs, on world-wide Computing Grids or on virtual Clouds of CPUs. And this is good news as far as EAs and EMOAs are concerned, as their coarse-grain parallelization is rather straightforward, and does not require to re-think the algorithm from the very beginning. Several models of parallel EAs have been proposed (see e.g. [3]), differing in the grain of the parallelization. In the master-slave model, a master node distributes fitness evaluations to slave nodes, and performs all evolutionary operations (selection, variation operators), and is hence identical, algorithmically speaking, to the sequential algorithm. In the Island models, each node runs a standard EA with a generally small population size, and the different nodes exchange individuals, called migrants, from time to time. In the totally distributed model, each node contains one (or very few) individuals, and the selection and crossover operations are performed amongst neighboring nodes. Note that both the Island model and the totally distributed model can have very different topologies for the different nodes, while the master-slave model requires a star-shaped topology.

The master-slave model is by far the simplest one to set up, and as such has been widely used in all application domains. Furthermore, because all evolutionary routines are performed in the master node, identically to the sequential case, it can be extended to the multi-objective case without any additional algorithmic effort. On the opposite, because EMOAs require some global synchronization at the selection step (see Section II), the island model and the totally distributed model need to be specifically adapted to the multi-objective context. For instance, a Parallel approach of NSGA-II has been proposed in [4], that is based on the "guided domination approach": each of the participating processors is assigned the task of finding only a particular portion of the Pareto set. A geometrical approach (the cone separation) is proposed in [5], that subdivides the search space in several regions. In order to have each processor focus on a specific region, the borders of each region are treated as constraints. In [6], a complex mechanism is designed that handles multiple populations across heterogeneous processors, based on mobile computing agents. However, most works in parallel MOEAs have focused on the master-slave model for its simplicity – and so does this paper.

The simplest implementation of the master-slave model uses the standard *generational* algorithm, in which the selection-variation steps only take place after all individuals of a given generation have been evaluated by the slaves. In the case where all evaluation require the same CPU time on all available processors, the population is uniformly distributed among the

slaves, and the speedup is exactly the number of processor, aka the *homogeneous* case, at least if the transmission time of the data over the network is small compared to the CPU cost of one evaluation [3]. However, in real-world problems in particular, the CPU costs of fitness evaluations is rarely homogeneous, be it because of the non-linearity of the simulation, or of the heterogeneity of the available computing resources. When using the generational model, many CPUs will remain idle while waiting for the longest evaluations. A well-known solution is to use the *steady-state* algorithm: in the sequential case, the steady-state algorithm generates one offspring, evaluates it, and replaces it in the population by removing on of the parents, generally based on some reverse tournament [7]. In the parallel heterogeneous case, one offspring is sent to a slave as soon as it is available, and the offspring are inserted in the population on a 'first-come first-served' basis (see Section II for more precise descriptions). Due to the heterogeneity of the fitness evaluation run times, individuals are inserted in the population in an *asynchronous* way, i.e. in a different order than when they have been generated. Contrary to the generational case, the asynchronous steady-state parallel algorithm is hence quite different than its sequential version.

The debate between steady-state and generational survival selections has been going on for long. In a general real-world framework, [8] argues that steady-state performs very often better than generational, and even more so in a multi-objective optimization context. In [9], another comparison between steady-state and generational NSGA-II is proposed on a real case study. However, to the best of our knowledge, no systematic comparison has been made between the asynchronous steady-state and the generational algorithms to analyze the consequence of heterogeneous evaluation computational costs.

This paper is motivated by a case study of MOEA applied to the (real-world) problem of the optimization of a Diesel engine (from the shape of the combustion chamber itself to the injection conditions) in order to decrease polluting emissions while preserving the power [10]. The computation of the different objectives relies on a time-consuming simulation of combustion, for which even a very simplified model requires on average 3 hours of CPU. Furthermore, this is a typical case where the actual cost of a fitness evaluation depends on the data. But after running both generational and asynchronous steady-state MOEAs, and confirming the benefits of the latter in the heterogeneous context, it turned out that the asynchronous algorithm was not only faster, never leaving a processor idle, but also identified more points on the Pareto front than its generational counterpart. The goal of this paper is to experimentally study the influence of heterogeneity of fitness evaluation costs on the performance of some asynchronous EMOAs (namely NSGA-II [11] and MO-CMA-ES [12]). Artificial benchmarks are proposed, where the localization of the heterogeneity of the evaluation costs in the search space can be controlled.

The remainder of this paper is structured as follows. Section II discusses the different master-slave ways of parallelizing MOEAs, and rapidly introduces NSGA-II and MO-CMA-ES that have been used in this work. Section III is devoted to a quick introduction of the case study that motivated this work, the optimization of Diesel combustion using an asynchronous NSGA-II algorithm. Section IV describes the experimental test-bench that was set up to emulate heterogeneous fitness computational costs on standard (and non-costly) test functions. The comparative results are discussed in Section V, and as usual Section VI summarizes the contribution of this work and suggests directions for future research.

II. Parallel Master-Slave MOEAs

This Section briefly describes the two MOEAs that will be concerned in this work, namely NSGA-II and MO-CMA-ES. Both the generational and steady-state versions will be surveyed. Note that many different MOEAs have been proposed since the very first one in the mid 80s [13]. All recent proposals are based on the idea of Pareto dominance: a solution x Pareto-dominates a solution y if x is better than y on all objectives, and strictly better on at least one objective. The algorithm then proceeds like a single-objective EA, except for the selection procedures (both parental and survival), that are replaced by a Pareto-based selection that consists of two hierarchical criteria: the first one uses Pareto dominance; however, because Pareto dominance does not induce a total order on the search space, a secondary criterion is necessary, that enforces the dispersion of the solutions over the Pareto front.

Among the many MOEAs that have been proposed in the literature, only two have been considered in this work, namely NSGA-II (Non-dominated Sorting Genetic Algorithm), proposed in the early 90s [11], still considered today as one of the state-of-the-art MOEA for its robustness across a variety of application domains, and the more recent MO-CMA-ES (Multi-Objective Covariance Matrix Adaptation Evolution Strategy) [12], that transposes to the multi-objective context the qualities of the single-objective CMA-ES [14]. Both will be briefly described in turn, in their original sequential versions. Their parallelization will then be discussed.

A. NSGA-II

The original (generational) NSGA-II algorithm is a Pareto-based MOEA that uses Non-Dominated Sorting as a first selection criterion, and Crowding Distance as secondary diversity-preserving criterion.

Non-Dominated Sorting is based on the "Pareto rank" of individuals in a given population: the non-dominated individuals are given rank 1 and removed from the population. The non-dominated individuals of the remaining of the population are given rank 2, and the process continues until all individuals have a Pareto rank. Crowding Distance considers the objective sequentially. The individuals are sorted according to objective i, and the partial crowding distance of the individual that has rank r with respect to objective i is the difference between the values of objective i of individuals with ranks $r-1$ and $r+1$. The Crowding Distance of an individual is then the sum over

all objectives of its partial crowding distances. The comparison between two individuals is as follows: if their Pareto ranks are different, the smallest one is preferred, favoring progress toward the Pareto front; If they are equal, the one with largest Crowding Distance is preferred, favoring diversity.

Using this comparison criterion, both generational and steady-state [7] versions of NSGA-II can be easily described. Both use a population of size P, and deterministic tournament of user-defined size T as parental selection. The generational NSGA-II uses a standard $P+P$ evolution engine: P offspring are generated, and the best of the P parents + P offspring, according to the comparison described above become the parents of next generation. The steady-state version of NSGA-II uses a standard $P+1$ steady-state evolution engine: 1 offspring is generated, and replaces in the population the "winner" of a reverse deterministic tournament of user-defined size T'.

B. MO-CMA-ES

The MO-CMA-ES algorithm proposed by Igel et al [12] extends the single-objective CMA-ES [14]. Like NSGA-II, it uses the Non-Dominated Sorting as main comparison criterion between solutions. The hypervolume-indicator [15] is uses as secondary comparison criterion.

All versions of the algorithm consider μ parents, each one being viewed as $(1+1)$-CMA-ES algorithm. The generational version is then a $(\mu+\mu)$ algorithm without parental selection and using only Gaussian mutation: each of the μ parents generates one offspring, and the μ best (according to the sorting criterion described above) of the $2*\mu$ parents plus offspring become the parents of the next generation. The mutation parameters of the selected offspring are updated following standard rank-one update for the covariance matrix, and a specific update rule for the step-size (see [12] for all details).

A steady-state version of MO-CMA-ES can be easily obtained by considering generating a single offspring and inserting it in the population immediately after evaluation, i.e., as for NSGA-II, using a $\mu+1$ steady-state evolution engine. Two possible variants were proposed in [16]: in the $(\mu+1)$ algorithm, the parent is selected uniformly in the whole population, while in the $(\mu_{\prec}+1)$ version, the parent is selected only amongst non-dominated individuals in the population. Further improvement were recently obtained by considering a tournament for parental selection rather than uniform selection [17]. However, only the $(\mu_{\prec}+1)$ will be considered in this work.

C. Parallel MOEAs

The algorithms described so far are sequential algorithms. In particular, a single CPU computes in turn all evaluations. In the steady-state versions of the algorithms, this means that offspring are inserted back in the population in the same order than they have been generated. When parallelizing these algorithms in a master-slave context, the master node takes care of all evolutionary operations (initialization, selection and variation), but sends out to its slaves all evaluations. In the homogeneous case, i.e., when all evaluations have the same run time, all algorithms behave the same than their sequential versions. However, the situation changes in the heterogeneous case, when some evaluations require a lot more computation time than others. The generational algorithms have to wait for all individuals to be evaluated before going on to next generation, and some nodes might stay idle for a long time, waiting for the slowest evaluations to complete. But the outcome of the parallel algorithm will be the same than that of the sequential one, only taking longer time to complete. In order to make better use of the available nodes, the steady-state algorithms can be slightly modified and made *asynchronous*, i.e. offspring are sent out for evaluation, and inserted back in the population on a first-come first-served basis. Thus no node stays idle for a long time, which was the primary motivation of asynchronous parallel algorithm.

Steady-state selection scheme has been applied in several MOEAs in the literature: the ϵMOEA [18] is based on the ϵ-dominance concept and uses steady-state selection scheme and archive update strategy. SMS-EMOEA proposed by Beume et al. [19] is a steady-state MOEA that uses the hypervolume criterion as the secondary selection criteria – but these works address only the sequential context. The parallel versions of these algorithms have been studied in detail in [9], on a real-world application, and indeed the asynchronous version was demonstrated to perform better, but the main concern in this work was that of the total elapsed time for large hardware platform (e.g., peer-to-peer networks). And heterogeneity in fitness computations came mostly from the hardware, and in particular did not depend on the individual being evaluated.

Because indeed, the behavior of the parallel algorithm might be different than that of the sequential one, depending on the distribution of the evaluation cost over the search space: a uniform distribution (i.e. the evaluation cost does not depend on the individual being evaluated) will most probably result in a behavior of the parallel algorithm similar to that of the sequential one: offspring are inserted back in the population in random order, and differences, averaged over the whole run, are likely to cancel out. However, if the evaluation costs do depend on the characteristics of the individuals, some region of the search space might be less explored than others, and some parts of the Pareto Front discovered later ... or never discovered at all.

Such phenomenon does take place in real world problems, as illustrated by the case-study in Section III, and this is the motivation for this work.

III. Optimization of Diesel Combustion

This section presents the real world problem on which a master-slave approach based on NSGA-II has been applied, first describing the optimization problem itself, and analyzing the obtained results in terms of computing time per fitness evaluation.

A. The problem

Due to environmental regulations, that are becoming very strict in Europe, the development of efficient automotive engines with low fuel consumption and low pollutant emissions becomes a hard task for engine designers. More specifically, three objectives are to be minimized, namely: NOx, $HCCO$, and Fuel consumption, and they are naturally in conflict: this problem pertains to multi-objective optimization, and it was decided to use NSGA-II [11] to solve this problem because of its well-known robust efficiency.

All the objectives can be computed through a multidisciplinary simulation of the combustion. Simulating the complete 3D model requires a huge amount of time – around 3 days on a recent single-core computer. Furthermore, the complexity of the phenomena that occur in the combustion chamber requires a very fine discretization in the numerical model, which in turn increases considerably the cost of the simulations. Hence a simplified model is used here, that does not take into account the exact geometry of the combustion chamber, and thus only involves the 10 decisions variables that control the combustion process, i.e., the injection parameters, and the air/fuel mixing parameters (more details can be found in [10]). The simulation time is reduced to ...a few hours. However, depending on the parameters, and because highly non-linear phenomena are involved in combustion models, the actual simulation run times vary a lot depending on the parameters, from 1.5 to 24 hours, as can be seen on Figure 1 for 40 random instances of the parameters. In order to cope with this complexity, it was decided to use a 40-CPUs cluster, and hence to use parallel versions of NSGA-II with population size 40. Furthermore, in order to keep all experiments within a reasonable range of CPU costs, all single fitness evaluations were stopped after roughly 8 hours, based on a human decision: the simulation is an iterative procedure; looking at the results after 8 hours, the runs that look unpromising with respect to convergence of the iterative procedures were immediately stopped, while more promising runs were continued up to a maximum of 12 hours.

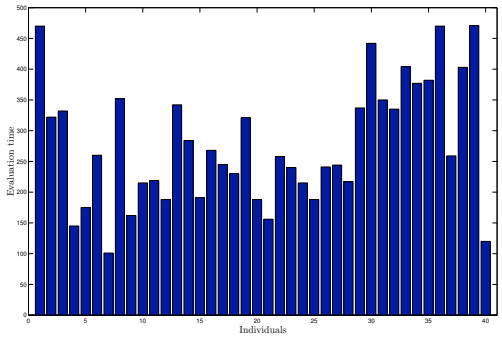


Fig. 1. Run time (in minutes) of fitness computations for 40 random individuals. All runs are stopped after 8 hours, to avoid overly long computations.

B. Experimental Results

Two variants of parallel NSGA-II were experimented with: the standard parallel generational algorithm, and the asynchronous steady-state version, both with population size 40. Both algorithms were run for 1600 evaluations, regardless of their durations (i.e. 40 generations for the generational variant). For obvious reasons, the results of a single run are reported here, as an illustration of the effects of heterogeneity on multi-objective costs ... and results.

Table I shows the run time required along evolution, at different numbers of function evaluation, for both generational and asynchronous algorithms. The total optimization cost for the generational NSGA-II is above 17 days, while that of the asynchronous NSGA-II is less than 10 days. As expected, the usage of CPU is much more efficient with the asynchronous steady-state algorithm than for the generational one.

TABLE I
COMPARISON OF RUN TIMES FOR DIFFERENT NUMBERS OF EVALUATIONS

# evaluations	Run time (hours)	
	Generational Parallel	Asynchronous Steady-State
200	39,5	26.52
400	81.23	52.69
600	130.35	82.54
800	182.6	110.98
1000	253.32	138.13
1200	304.16	164.2
1400	351.1	200.2
1600	410.3	236.2

More interestingly, Figure 2 and 3 display some 2D projections of the Pareto fronts in the space "HCCO-Fuel consumption" vs "NOx-Fuel consumption". Both optimizations are comparable in terms of quality of convergence. However, some very efficient solutions in terms of NOx values (with of course poor HCCO values, as both objectives are contradictory) have been discovered by the asynchronous optimization, and not in the generational case. A closer look revealed that these solutions represent individuals with large evaluation time, close or over the 8 hours threshold. As explained above, these individuals are likely to be killed in the generational approach. Thus, they do not appear in the Pareto front of the generational optimization although they belong there.

These experiments, beside having provided interesting solutions in terms of Engine Optimization [10], also demonstrated that the heterogeneity of fitness computational costs was a clear argument for using asynchronous parallel algorithms rather than generational ones, in order not to waste any computational power. Furthermore, these results raise the issue of possible benefits of asynchronous steady-state algorithms compared to the standard generational approach. It is however not clear whether the "missing" solutions in the generational Pareto front would have been discovered by the generational approach without the 8-hours threshold. The experiments presented in the rest of this work aim at answering such questions within an artificial framework to avoid the intractable computation run times of the Engine Optimization problem.

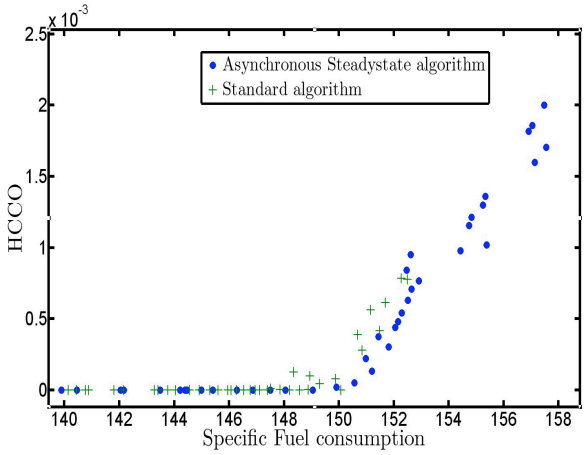


Fig. 2. 2D Pareto Front: HCCO vs Fuel Consumption

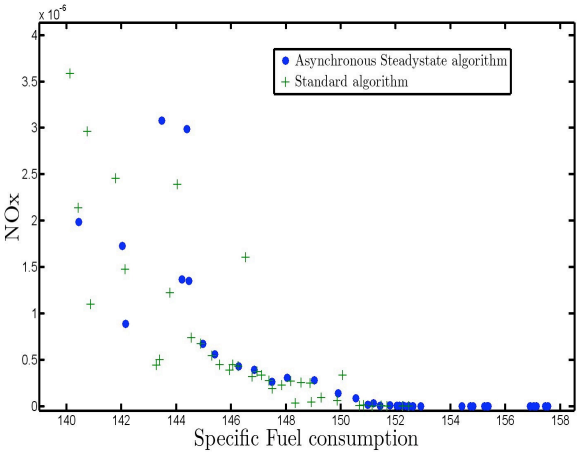


Fig. 3. 2D Pareto Front: NOx vs Fuel Consumption

IV. Artificial Heterogeneous Testbench

The baseline for the proposed experiments will be analytic test functions that have been widely used in the EMO community, namely the ZDT [20] and IHR [12] test suites. Those test problems are very easy and fast to compute, and the "cost" of the evaluation will hence be introduced artificially . . . and it will hence be possible to carefully tune model of the evaluation cost.

A. Implementation

The implementation of an evaluation cost model is straightforward: the algorithm maintains an additional queue that contains the evaluated offspring in the order they will be inserted back in the population at the end of the steady-state loop. First, and for simplicity reasons, the initial population is evaluated at once, and even in the steady-state variants, the heterogeneity of the evaluation costs is not taken into account there. Note that preliminary experiments showed no significant difference whatsoever when the initial population was gradually filled with individuals as they returned from evaluation. The main loop of the heterogeneous steady-state algorithms start by selecting a parent (with the parental selection at hand), and "sends" it for evaluation (i.e., evaluates it). Depending on the values of the objectives, the "run time" of this evaluation is computed from the evaluation cost model. The expected return time of the offspring is the current time + the evaluation run time. The offspring is then inserted back into the waiting queue depending on this return time (the waiting queue contains all evaluated offspring, sorted by return time). The offspring that is sitting on top of the waiting queue is then inserted in the population, using the survival selection at hand.

B. The cost models

The baseline model assumes that there is no correlation whatsoever between the evaluation cost and the individual being evaluated, and covers the case where the only source of heterogeneity comes from the available resources. Practically speaking, using the baseline model amounts to inserting the new offspring randomly in the waiting queue.

Two other models for this additional cost will be used here. The first model assumes that the evaluation cost increases inversely proportionally to the distance to the Pareto front. This is the case for instance when some increased accuracy of results is needed as search nears the true Pareto front (e.g., requiring more iterations, or finer mesh, or . . .). The second model is loosely inspired by the real-world application of Engine Optimization described in Section III, and assumes that all solutions in some rectangle of the objective space take longer to compute than elsewhere.

For the first model, the ε-indicator is used to estimate the 'distance' to the Pareto front, that is known analytically for all test functions ZDT and IHR. The ε-indicator [21] gives the factor by which a set A is worse than another set B with respect to all objectives. The additive version that is considered here is defined, for 2 sets A and B from the search space, by $inf\left\{\varepsilon; (\forall z^b \in B)(\exists z^a \in A)(z_i^a \leq z_i^b + \epsilon)_{i=1,\ldots,N}\right\}$. The evaluation cost of a newborn offspring is set to the inverse of the square of ε-indicator, between the offspring in search space and the true Pareto front for the problem at hand.

For the second model, all individuals have the same evaluation cost, except those in a predefined area of the objective space, defined by bounds on the values of the different objectives, that are penalized proportionally to their objectives values. This allows us to set a higher evaluation cost around any part of the Pareto Front, inspired by what happened for the real-world problem described in Section III.

V. Experiments

A. Setting and Plots

As already mentioned, experiments were conducted on the ZDT [20] and IHR [12] test suites, more precisely on their 1-3;6 instances. All reported results are averages (or aggregations) over 30 independent runs. All runs were stopped after 50000 function evaluations. The sizes of the population and the waiting queue for evaluated offspring (see Section

IV-A) were set to 100, thus simulating the parallelization on a 100-nodes cluster or grid.

In a first series of experiments, 4 variants of each algorithm are compared: for algorithm X (NSGA-II or MO-CMA-ES), the base generational version is termed X_{gen}, the synchronous steady-state version $X_{steady-state}$, and results of the asynchronous steady-state variant are indexed by the evaluation cost model that is being used (see Section IV): the heterogeneous case with random model is termed X_{random} and the heterogeneous case where evaluation cost increases close to the Pareto front, also called in the following ε-penalized, is denoted $X_{epsilon}$.

The performance measurement considered for this series of experiments is the Hypervolume indicator [15], known to be consistent with Pareto dominance. For real-world problems, with costly evaluation, the most important limiting factor computation-wise is the total elapsed time. It can be computed here based on the evaluation cost model for the generational and ε-penalized methods (for the generational method, after all offspring from one generation have been generated, the one with the longest run time determines the run time of the generation). But for the purpose of comparison, the total computational effort, classically measured here in terms of number of function evaluation, allows us to compare different algorithmic settings.

A second series of experiments dealt with the second evaluation cost model, that penalizes a precise region of the objective space. As expected, this penalization hinders the search in the corresponding region of the search space, and the penalized region of the Pareto front is discovered very late - if discovered at all. In such context, global indicators like the hypervolume are not useful to analyze and compare the different algorithms. Only several snapshots of the current Pareto Front could possibly illustrate the different behaviors of the algorithms - and these cannot be shown due to the space limitations. A further technical report will detail these results, but they will not be reported on here. Note that the global picture (hypervolume plots similar to Figures 4 and 7) are very similar to the ones obtained using the ε-cost model.

B. Results

Tables II and III summarize the final results obtained on the 8 instances by the variants of NSGA-II and MO-CMA-ES respectively, reporting the average hypervolume indicator, as well as the statistical significance of the observed differences.

Some clear results appear for NSGA-II on ZDT functions. First, NSGA-II$_{steady-state}$ outperforms all other variants on ZDT1 and ZDT2. On ZDT3, NSGA-II$_{steady-state}$ and the asynchronous versions on either evaluation cost model (NSGA-II$_{random}$ and NSGA-II$_{epsilon}$) are equivalent, and significantly outperform the generational version. However, NSGA-II$_{epsilon}$ algorithm is also the slowest in terms of convergence speed, due to the fact that the good individuals are expensive to evaluate. However, the asynchronous algorithm finally catches up with the generational one after some number of evaluations (30000 or ZDT1 and ZDT3, 45000 for ZDT2). This confirms the results from [8], [9] demonstrating that steady-state algorithms performed often better than generational ones in several contexts, and in particular in a multi-objective setting. The situation is typically visible on Figure 4: the right plot shows the delay for NSGA-II$_{epsilon}$, whereas the left plot illustrates the well-known benefit of asynchronous steady-state selection, in terms of elapsed time.

Results obtained on IHR problems with NSGA-II show that NSGA-II$_{epsilon}$ outperforms other algorithms on IHR1, IHR2, and is equivalent on IHR3 and IHR6. Our interpretation of this phenomena is the following. On IHR problems, a premature convergence toward a small part of the Pareto front in observed with NSGA-II$_{gen}$ (Figure 5-left). With NSGA-II$_{epsilon}$ algorithm, the best individuals are penalized, and paradoxically, this prevents such premature convergence, and the algorithm succeeds in finding more points on the Pareto front (Figure 5-center). Note that NSGA-II$_{random}$ does also discover a larger part of the Pareto front (Figure 5-right). However, this can be said from only a few runs, whereas the plots of Figure 5 are aggregations of 30 runs. Table II shows that indeed, NSGA-II$_{gen}$ is significantly better than NSGA-II$_{random}$. The difference can even be more critical when the Pareto front is discontinuous, as for IHR3: Figure 6 shows that indeed, complete components of the Pareto front can be missed by one algorithm or the other.

Regarding MO-CMA-ES, all steady-state variants outperform the generational one on all functions except ZDT6. Again, this confirms previous results [16]. Furthermore, the synchronous and the asynchronous (with both cost models) variants perform quite similarly in terms of convergence speed. For most test instances, MO-CMA-ES$_{epsilon}$ is the slowest, similarly, and for the same reasons, that what can be seen or NSGA-II on Figure 4. A sligthly different picture emerges on IHR1 problem, where, unexpectedly, the random model penalizes the asynchronous algorithm, as can be seen on Table III, and on Figure 7-left. Nevertheless, hypervolume versus elapsed-time plots (such as Figure 7-right) show that the asynchronous steady-state algorithm requires much less computational time than the generational version, while providing a better solution quality (except for ZDT6).

VI. Discussion and Conclusion

This paper has investigated the influence of the distribution of heterogeneity of evaluation computational cost on the results of parallel EMOAs in a master-slave asynchronous setting, motivated by a real-world case study dealing with the optimization of the combustion in a Diesel engine. Investigations have been conducted for two well-known state-of-the-art algorithms, NSGA-II and MO-CMA-ES, for which we proposed a test-bench based on a classical set of test functions (ZDT and IHR) but using any user-defined model for the cost of evaluations. Experiments have been made with two cost models to simulate heterogeneity on some distributed system (e.g. grid or cluster).

Experiments using the first cost model (run time increases as solutions near the Pareto front) have demonstrated that the asynchronous algorithm outperforms the generational one in both quality of convergence and global run time, for all considered cost models. Smaller computational cost was of course expected, because the asynchronous steady-state algorithm ensures an interrupted use of processors on multi-processor systems, while generational algorithms need to synchronize at each generation. As for the quality of the results, in terms of hypervolume indicator, the better results obtained by asynchronous algorithm come from its steady-state selection, almost independently of the evaluation cost model. However, a close look at the dynamic of the optimization suggests that the higher evaluation cost close to the Pareto front slows down the convergence, and thus improves the quality of the results, as demonstrated on IHR functions.

The second cost model was inspired by the case study of the Diesel engine, and assumes that the evaluation cost is high in a specific region of the objective space including part of the Pareto front. Based on previous results on the case study, where the asynchronous algorithm had found solutions that the generational version did not discover, our hope was that the asynchronous algorithm would be able to repeatedly cover a larger part of the Pareto front than the generational one. Though this did sometimes happen, the results in this direction were not robust enough to support any strong claim. Further work will nevertheless try to address this issue, and our long-term goal is come up with a new point of view on the exploitation vs exploration dilemma. Ultimately, we should be able to identify certain states of evolution, and to somehow slow down convergence and favor coverage when necessary.

TABLE II
MEDIAN RESULTS OF HYPERVOLUME INDICATOR OVER 30 TRIALS AFTER 50000 EVALUATIONS WITH NSGA-II VARIANTS. SUPERSCRIPTS INDICATE SIGNIFICANT DIFFERENCES WITH THE CORRESPONDING COLUMN ACCORDING TO A KRUSKAL-WALLIS TEST WITH $\alpha = 0.01$.

Test	hypervolume indicator			
	NSGAII$_{gen}$	NSGAII$_{ss}$	NSGAII$_{eps}$	NSGAII$_{rand}$
ZDT1	0.001923	$0.000123^{1,3,4}$	$0.000228^{1,4}$	0.000231^{1}
ZDT2	0.001865	$0.000431^{1,3,4}$	0.001121^{1}	0.000853^{1}
ZDT3	0.001265	0.000199^{1}	0.000422^{1}	0.000377^{1}
ZDT6	4.072532	4.060275	4.092183	4.080733
IHR1	0.572032^{4}	0.580593^{4}	$0.423853^{1,2,4}$	0.825886
IHR2	9.929486	11.652031	$5.091023^{1,2}$	$6.299612^{1,2}$
IHR3	1.979927	2.041639	1.796320	1.975204
IHR6	27.96567	28.82385	30.33832	28.46332

TABLE III
MEDIAN RESULTS OF HYPERVOLUME INDICATOR OVER 30 TRIALS AFTER 50000 EVALUATIONS WITH MO-CMA-ES VARIANTS. SUPERSCRIPTS INDICATE SIGNIFICANT DIFFERENCES WITH THE CORRESPONDING COLUMN ACCORDING TO A KRUSKAL-WALLIS TEST WITH $\alpha = 0.01$.

Test	hypervolume indicator			
	MOCMA$_{gen}$	MOCMA$_{ss}$	MOCMA$_{eps}$	MOCMA$_{rand}$
ZDT1	0.000223	0.000078^{1}	0.000065^{1}	0.000074^{1}
ZDT2	0.000223	0.000088^{1}	0.000073^{1}	0.000088^{1}
ZDT3	0.000118	0.000045^{1}	0.000041^{1}	0.000041^{1}
ZDT6	2.617498	2.617398	2.617390	2.617328
IHR1	0.279109	$0.173253^{1,3,4}$	0.191495	0.323821
IHR2	10.480423	10.474721	10.476632	10.480034
IHR3	2.111123	1.730129^{1}	1.556394^{1}	1.641038^{1}
IHR6	12.572234	$11.34743^{1,3,4}$	14.875943	13.075328

REFERENCES

[1] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK: Wiley, 2001.

[2] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[3] E. Cantu-Paz, *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Pub, 2000.

[4] K. Deb., P. Zope, and A. Jain, "Distributed computing of pareto-optimal solutions with evolutionary algorithms," in *Evolutionary Multi-objective Optimization – EMO'03*, C. F. et al., Ed. LNCS 2632, Springer Verlag, 2003, pp. 534–549.

[5] J. Branke., H. Schmeck, and K. Deb, "Parallelizing multi-objective evolutionary algorithms: Cone separation," in *IEEE Congres on Evolutionary Computation 2004*, 2004, pp. 1952–1957.

[6] V. G. Asouti and K. C. Giannakoglou, "Aerodynamic optimization using a parallel asynchronous evolutionary algorithm controlled by strongly interacting demes," *Engineering Optimization*, vol. 41, no. 3, pp. 241–257, March 2009.

[7] G. Syswerda, "A study of reproduction in generational and steady state genetic algorithm," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. Morgan Kaufmann, 1991, pp. 94–101.

[8] D. Chafekar, J. Xuan, and K. Rasheed, "Constrained multi-objective optimization using steady state genetic algorithms," in *Genetic and Evolutionary Computation*, Cantú-Paz, Erick et al., Ed. Springer Verlag, 2003, vol. 2723, pp. 201–201.

[9] J. Durillo, A. Nebro, F. Luna, and E. Alba, "A Study of Master-slave Approaches to parallelize NSGA-II," in *Proc. IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1–8.

[10] M. Yagoubi, L. Thobois, and M. Schoenauer, "An Asynchronous Steady-state NSGA-II Algorithm for Multi-Objective Optimization of Diesel Combustion," in *Proc. 2nd Intl Conf. on Engineering Optimization – EngOpt'2010*, H. Rodrigues et al., Ed., 2010.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[12] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 1, pp. 1–28, 2007.

[13] D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms*, J. J. Grefenstette, Ed., 1985.

[14] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[15] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[16] C. Igel, T. Suttorp, and N. Hansen, "Steady-state selection and efficient covariance matrix update in MO-CMA-ES," in *EMO'07*, S. Obayashi et al., Ed. LNCS 2632, Springer Verlag, 2007, pp. pp.171–185.

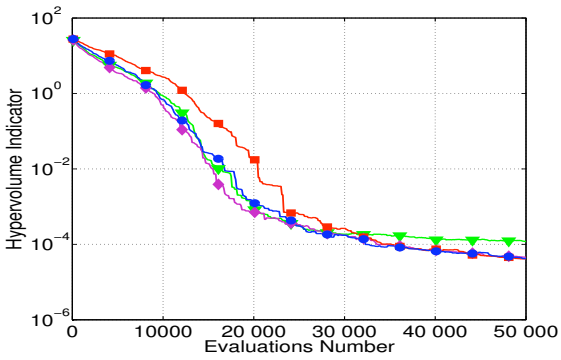
[17] I. Loshchilov, M. Schoenauer, and M. Sebag, "Not all parents are equal for mo-cma-es," in *Proc. EMO'2011*, ser. LNCS, Ricardo H. C. Takahashi et al., Ed. Springer Verlag, 2011, to appear.

[18] K. Deb, M. Mohan, and S. Mishra, "Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions," in *EMO'03*, C. Fonseca et al., Ed. LNCS 2632, Springer Verlag, 2003, pp. 222–236.

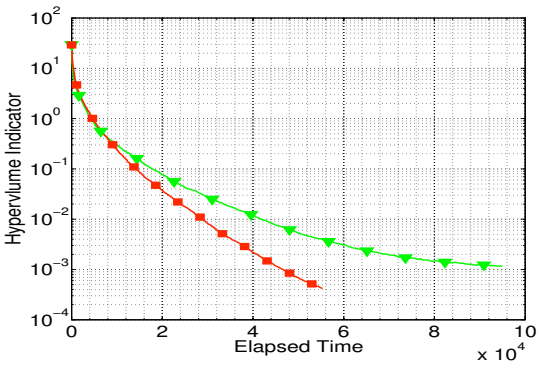
[19] N. Beume, B. Naujoks, and M. Emmerich, "Sms-emoa: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[20] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[21] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, "Performance Assessment of Multiobjective Optimizers: An Analysis and Review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

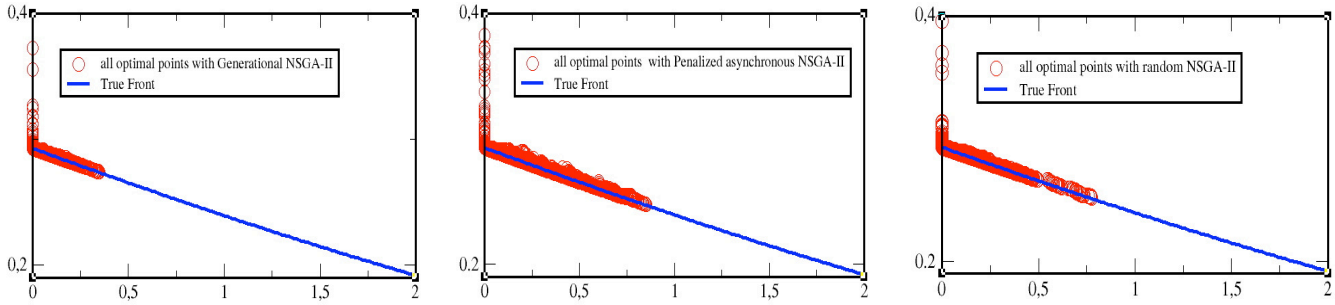


All 4 variants, vs # evaluations



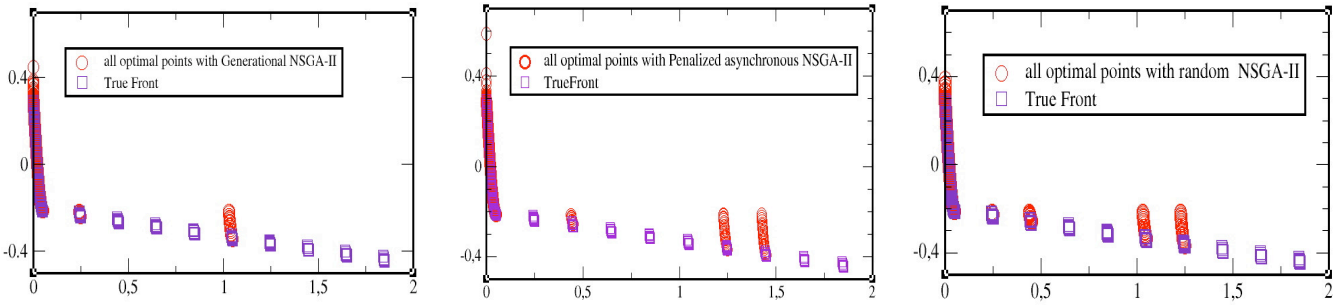
Generational and ε-penalized, vs elapsed time

Fig. 4. Evolution of average hypervolume indicator for NSGA-II on ZDT3. ▼ = generational; ♦ = steady-state; ■ = ε-penalized; ● = random.



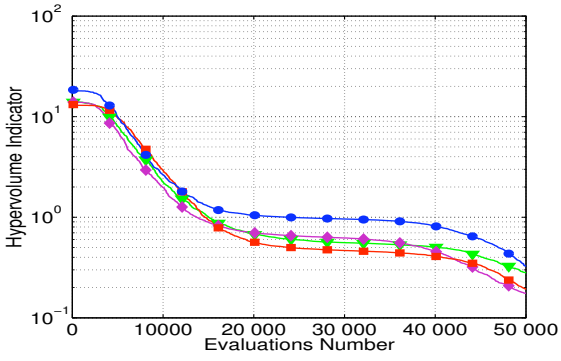
(a) NSGA-II$_{gen}$ (b) NSGA-II$_{epsilon}$ NSGA-II$_{random}$

Fig. 5. Merged approximate Pareto Fronts from 30 runs with 50000 evaluations for different versions of NSGA-II on IHR1 problem

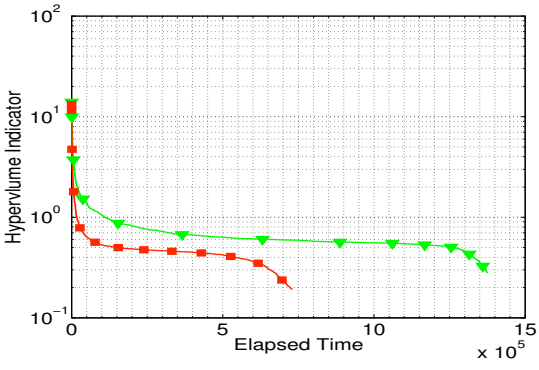


(a) NSGA-II$_{generational}$ (b) NSGA-II$_{epsilon}$ NSGA-II$_{random}$

Fig. 6. Merged approximate Pareto Fronts from 30 runs with 50000 evaluations for different versions of NSGA-II on IHR3 problem



All 4 variants, vs # evaluations



Generational and ε-penalized, vs elapsed time

Fig. 7. Evolution of average hypervolume indicator for MO-CMA-ES on IHR1. ▼ = generational; ♦ = steady-state; ■ = ε-penalized; ● = random.